Description Logics e

Semantic Web

Sommario

- Predicati one-of e fills
- Motivazioni
- Verso il semantic web
- OWL

Descrizioni con nomi di individui: one-of

"set" o **one-of** si scrive

$$\{a_1, \ldots, a_n\}$$

dove $a_1, \ldots, a_n$ sono individui

$$\{a_1, \ldots, a_n\}^{\mathcal{I}} = \{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$$

Esempio: $\{CHINA, FRANCE, RUSSIA, UK, USA\}$.

$\{a_1, \ldots, a_n\}$ e $\{a_1\} \sqcup \cdots \sqcup \{a_n\}$ sono equivalenti

Descrizioni con nomi di individui: fills

$$R : a,$$

R è un ruolo e a un individuo.

$$(R : a)^{\mathcal{I}} \;=\; \{d \in \Delta^{\mathcal{I}} \mid (d, a^{\mathcal{I}}) \in R^{\mathcal{I}}\}$$

$R : a$ è la classe di individui che hanno a come "R-filler"

$R : a$ e $\exists R.\{a\}$ sono equivalenti

Fills consente di esprimere asserzioni sui ruoli:
$R(a, b)$ sse $a : (\exists R.\{b\})$.

Idea del semantic web

The World Wide Web, while wildly successful in growth, may be viewed as being limited by its reliance on languages such as HTML that are focused on **presentation** (i.e., text formatting) rather than **content**. Languages such as Xml do add some support for capturing the meaning of Web content (instead of simply how to render it in a browser). [...]
[The Semantic Web] is to be achieved by augmenting the existing layout information with semantic annotations that add descriptive terms to Web content, with the meaning of such terms being defined in **ontologies**.

Idea del semantic web

"L'idea di documenti comprensibili dalle macchine non richiede alcuna magica intelligenza artificiale che permetta alle macchine di comprendere le rimuginazioni degli umani. Indica solamente l'abilità delle macchine di risolvere problemi ben definiti attuando operazioni ben definite su ben definiti dati esistenti. Invece di chiedere alle macchine di comprendere il linguaggio della gente, ciò significa chiedere alla gente di fare uno sforzo extra"
Tim Berners-Lee, "What the Semantic Web can represent", 1998. http://www.w3.org/DesignIssues/RDFnot.html

Obiettivi

- migliorare l'efficienza e la precisione dei motori di ricerca.
- realizzare sistemi di catalogazione dei contenuti e delle relazioni tra le pagine di un particolare sito web.
- favorire la condivisione e lo scambio di informazioni tra agenti software intelligenti.
- aumentare l'accessibilità dell'informazione e l'integrazione di informazioni provenienti da sorgenti diverse.
- riunire in un unico documento logico collezioni di pagine web semanticamente correlate ma distribuite su pi siti.
- semplificare l'automazione di transazioni di tipo commerciale, aumentandone la sicurezza.
- ...

The $\mathcal{AL}$ family of DL

	Construct Name	Syntax	Semantics
$\mathcal{AL}$	...		
$\mathcal{C}$	concept negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
	transitive role	$R \in \mathbf{R}_+$	$R^{\mathcal{I}} = (R^{\mathcal{I}})^+$

Si definisce $\mathcal{S}$ la famiglia $\mathcal{ALC}$ + transitive roles.

Si indica con (D) la possibilità di gestire *datatypes*, cioè alcuni tipi standard come int, char,string...
Esempio: $codicefiscale(GIUSEPPE,'' sttgpp80m29'')$

La famiglia $\mathcal{S}$: costruttori di concetti

	Construct Name	Syntax	Semantics
$\mathcal{N}$	number restrictions	$\geqslant n\, P$	$\{x \mid \sharp\{y.\langle x,y\rangle \in P^{\mathcal{I}}\} \geqslant n\}$
		$\leqslant n\, P$	$\{x \mid \sharp\{y.\langle x,y\rangle \in P^{\mathcal{I}}\} \leqslant n\}$

Esempio:"Mamma con molti figli": $Female \sqcap \geq\ 3\ hasChild$

	Construct Name	Syntax	Semantics
$\mathcal{Q}$	qualifying number restrictions	$\geqslant n\, P.C$	$\{x \mid \sharp\{y.\langle x,y\rangle \in P^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \geqslant n\}$
		$\leqslant n\, P.C$	$\{x \mid \sharp\{y.\langle x,y\rangle \in P^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \leqslant n\}$

Esempio:"Mamma con almeno due figli ingegneri":
$Female \sqcap \geq\ 2\ hasChild.Engineer$

La famiglia $\mathcal{S}$: costruttori di concetti

	Construct Name	Syntax	Semantics
$\mathcal{O}$	nominal	o	$o^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ with $\sharp\{o^{\mathcal{I}}\} = 1$

Esempio:"Chi lavora di lunedì": $Person \sqcap \exists\ works.MONDAY$

	Construct Name	Syntax	Semantics
$\mathcal{F}$	feature	$u_1 \doteq u_2$	$\{x \mid \exists b \in \Delta^{\mathcal{I}}.u_1^{\mathcal{I}}(x) = b = u_2^{\mathcal{I}}(x)\}$
	(dis)agreement	$u_1 \neq u_2$	$\{x \mid \exists b_1, b_2 \in \Delta^{\mathcal{I}}.u_1^{\mathcal{I}}(x) = b_1 \neq b_2 = u_2^{\mathcal{I}}(x)\}$

Gli u_i sono un sottoinsieme dei ruoli detti funzionali, cioè per cui vale la regola: $\forall a, b, c\ .\ \langle a, b\rangle \in R^{\mathcal{I}} \wedge \langle a, c\rangle \in R^{\mathcal{I}} \ \rightarrow\ b = c$

Esempio:"Gli autoctoni": $Person \sqcap (bornIn \doteq livesIn)$

La famiglia $\mathcal{S}$: Costruttori di ruolo

	Construct Name	Syntax	Semantics
$\mathcal{I}$	inverse role	R^-	$\{\langle x, y\rangle \mid \langle y, x\rangle \in R^{\mathcal{I}}\}$

Esempio: "I figli degli ingegneri"

$\exists\ hasChild^-.Engineer$

	Construct Name	Syntax	Semantics
$\mathcal{H}$	role hierarchy	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$

Esempio: Nei grafi, dato il concetto $Nodo$ e i ruoli $arco$ e $raggiungibile$ di tipo $Nodo \times Nodo$ si ha

$arco \sqsubseteq raggiungibile$

Verso il Semantic Web: RDFS

- RDFS: "Resource Description Framework" (w3c)
  - RDF è una specifica in XML per rappresentare contenuti ("risorse") sul web con semantica (cioè che portano significato)
  - RDFS: XML Schema è un vocabolario di termini per descrivere proprietà e classi di oggetti secondo RDF

Linguaggi per ontologie: DAML+OIL

OIL realizzò (1999) il primo linguaggio basato su Description Logics per la rappresentazione di contenuti sul web. DAML nello stesso periodo realizzò un linguaggio per agenti usando RDFS.
La loro fusione (DAML+OIL) costituisce il primo esempio di linguaggio per il semantic web. La capacità espressiva equivale a $\mathcal{SHIQ}(\mathcal{D})$.

	Construct Name	Syntax	Semantics
$\mathcal{H}$	role hierarchy	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
$\mathcal{I}$	inverse role	R^{-}	$\{\langle x, y\rangle \mid \langle y, x\rangle \in R^{\mathcal{I}}\}$
$\mathcal{Q}$	qualifying number restrictions	$\geqslant n\, P.C$	$\{x \mid \sharp\{y.\langle x, y\rangle \in P^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \geqslant n\}$
		$\leqslant n\, P.C$	$\{x \mid \sharp\{y.\langle x, y\rangle \in P^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \leqslant n\}$

Influenze nel design di OWL

Il progetto w3c OWL (Ontology Web Language) risente dell'influenza di

- formalismi già consolidati per la rappresentazione della conoscenza (DL)
- precendenti linguaggi per le ontologie (DAML+OIL)
- precenti linguaggi per il web semantico (RDFS)

OWL è una w3c recommendation dal 10 febbraio 2004.

OWL DL

- Specifica W3C per OWL
  http://www.w3.org/TR/owl-guide/
  http://www.w3.org/TR/owl-features/
- Non ha l'Unique Name Assumption.
- Equivalente a $\mathcal{SHOIN}(\mathbf{D})$. Ragionamento nel caso peggiore in NExpTime.

	Construct Name	Syntax	Semantics
$\mathcal{H}$	role hierarchy	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
$\mathcal{O}$	nominal	o	$o^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ with $\sharp\{o^{\mathcal{I}}\} = 1$
$\mathcal{I}$	inverse role	R^{-}	$\{\langle x, y\rangle \mid \langle y, x\rangle \in R^{\mathcal{I}}\}$
$\mathcal{N}$	number restrictions	$\geqslant n\, P$	$\{x \mid \sharp\{y.\langle x, y\rangle \in P^{\mathcal{I}}\} \geqslant n\}$
		$\leqslant n\, P$	$\{x \mid \sharp\{y.\langle x, y\rangle \in P^{\mathcal{I}}\} \leqslant n\}$

Sintassi OWL DL

Abstract Syntax	DL Syntax
Descriptions (C)	
A	A
`owl:Thing`	$\top$
`owl:Nothing`	$\bot$
`intersectionOf`($C_1 \ldots C_n$)	$C_1 \sqcap \ldots \sqcap C_n$
`unionOf`($C_1 \ldots C_n$)	$C_1 \sqcup \ldots \sqcup C_n$
`complementOf`(C)	$\neg C$
`oneOf`($o_1 \ldots o_n$)	$\{o_1\} \sqcup \ldots \sqcup \{o_n\}$
`restriction`(R `someValuesFrom`(C))	$\exists R.C$
`restriction`(R `allValuesFrom`(C))	$\forall R.C$
`restriction`(R `hasValue`(o))	$R : o$
`restriction`(R `minCardinality`(n))	$\geqslant n\, R$
`restriction`(R `minCardinality`(n))	$\leqslant n\, R$

Abstract Syntax	DL Syntax
`restriction(`U `someValuesFrom(`D`))`	$\exists U.D$
`restriction(`U `allValuesFrom(`D`))`	$\forall U.D$
`restriction(`U `hasValue(`v`))`	$U : v$
`restriction(`U `minCardinality(`n`))`	$\geqslant n\, U$
`restriction(`U `maxCardinality(`n`))`	$\leqslant n\, U$
Data Ranges (D)	
D	D
`oneOf(`$v_1 \ldots v_n$`)`	$\{v_1\} \sqcup \ldots \sqcup \{v_n\}$
Object Properties (R)	
R	R
`inv(`R`)`	R^-
Datatype Properties (U)	
U	U
Individuals (o)	
o	o
Data Values (v)	
v	v

Sintassi OWL DL 2

Abstract Syntax	DL Syntax
Class(A partial $C_1 \ldots C_n$)	$A \sqsubseteq C_1 \sqcap \ldots \sqcap C_n$
Class(A complete $C_1 \ldots C_n$)	$A \equiv C_1 \sqcap \ldots \sqcap C_n$
EnumeratedClass(A $o_1 \ldots o_n$)	$A \equiv \{o_1\} \sqcup \ldots \sqcup \{o_n\}$
SubClassOf(C_1 C_2)	$C_1 \sqsubseteq C_2$
EquivalentClasses($C_1 \ldots C_n$)	$C_1 \equiv \ldots \equiv C_n$
DisjointClasses($C_1 \ldots C_n$)	$C_i \sqcap C_j \subseteq \bot, i \neq j$
Datatype(D)	
ObjectProperty(R super(R_1)...super(R_n)	$R \sqsubseteq R_i$
domain(C_1)...domain(C_m)	$\geqslant 1\, R \sqsubseteq C_i$
range(C_1)...range(C_ℓ)	$\top \sqsubseteq \forall R.C_i$
[inverseOf(R_0)]	$R \equiv R_0^-$
[Symmetric]	$R \equiv R^-$
[Functional]	$\top \sqsubseteq \leqslant 1\, R$
[InverseFunctional]	$\top \sqsubseteq \leqslant 1\, R^-$
[Transitive])	$Tr(R)$

Abstract Syntax	DL Syntax
SubPropertyOf(R_1 R_2)	$R_1 \sqsubseteq R_2$
EquivalentProperties($R_1 \ldots R_n$)	$R_1 \equiv \ldots \equiv R_n$
DatatypeProperty(U super(U_1)...super(U_n)	$U \sqsubseteq U_i$
domain(C_1)...domain(C_m)	$\geqslant 1\, U \sqsubseteq C_i$
range(D_1)...range(D_ℓ)	$\top \sqsubseteq \forall U.D_i$
[Functional])	$\top \sqsubseteq \leqslant 1\, U$
SubPropertyOf(U_1 U_2)	$U_1 \sqsubseteq U_2$
EquivalentProperties($U_1 \ldots U_n$)	$U_1 \equiv \ldots \equiv U_n$
AnnotationProperty(S)	
OntologyProperty(S)	
Individual(o type(C_1)...type(C_n)	$o \in C_i$
value(R_1 o_1)...value(R_n o_n)	$\langle o, o_i \rangle \in R_i$
value(U_1 v_1)...value(U_n v_n))	$\langle o, v_i \rangle \in U_i$
SameIndividual($o_1 \ldots o_n$)	$\{o_1\} \equiv \ldots \equiv \{o_n\}$
DifferentIndividuals($o_1 \ldots o_n$)	$\{o_i\} \sqsubseteq \neg\{o_j\}, i \neq j$

Verbosity of OWL

Description Logic syntax:
$\texttt{Student} \equiv \texttt{Person} \sqcap \geqslant 1\, \texttt{enrolledIn}$

OWL syntax:

```
<owl:Class rdf:ID="Student">
   <owl:intersectionOf rdf:parsetype="Collection">
      <owl:Class rdfs:about="Person" />
      <owl:Restriction>
         <owl:onProperty rdf:resource="enrolledIn" />
         <owl:minCardinality rdfs:datatype="&xsd;Integer">
                1
         </owl:minCardinality>
      </owl:Restriction>
   </owl:intersectionOf>
</owl:Class>
```

OWL Lite

Le principali limitazioni di OWL Lite sono di non permettere:

- `Vincoli di cardinalità diversi da 0-1.`
- `Creazione di concetti enumerati (oneof).`
- `Specifica di concetti basati sull'esistenza di un particolare slot-filler.`
- `Creazione di concetti definiti.`

OWL Lite appartiene a $\mathcal{SHIF}(\mathbf{D})$. Ragionamento nel caso peggiore in ExpTime

	Construct Name	Syntax	Semantics
$\mathcal{H}$	role hierarchy	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
$\mathcal{I}$	inverse role	R^{-}	$\{\langle x, y\rangle \mid \langle y, x\rangle \in R^{\mathcal{I}}\}$
$\mathcal{F}$	feature	$u_1 \doteq u_2$	$\{x \mid \exists b \in \Delta^{\mathcal{I}}.u_1^{\mathcal{I}}(x) = b = u_2^{\mathcal{I}}(x)\}$
	(dis)agreement	$u_1 \neq u_2$	$\{x \mid \exists b_1, b_2 \in \Delta^{\mathcal{I}}.u_1^{\mathcal{I}}(x) = b_1 \neq b_2 = u_2^{\mathcal{I}}(x)\}$

OWL Full

- OWL Full contiene OWL DL ma esce al di fuori degli standard di DL.
- La principale introduzione è la possibilità di trattare un concetto come un individuo di un altro concetto.
- Le interrogazioni su un'ontologia OWL Full sono in generale indecidibili.

Ragionatori

DIG Interface: "Description Logics Implementers Group"
Interfaccia (API) basata su XML per comunicazione tra ragionatore e definizione di basi di conoscenza basata su Description Logics.
Questo permette di strutturare il ragionatore come un server in attesa di una KB, gli editor come client.

I piu comuni ragionatori

- Pellet
  - basato su java
  - open source
  - forti problemi di memoria
  - http://pellet.owldl.com/
- Racer
  - basato su lisp
  - commercial (con licenza d'uso per scopi di ricerca)
  - ottime prestazioni
  - http://www.racer-systems.com

- Kaon2

  - basato su java
  - free for non commercial use
  - non supporta datatype
    (ruoli basati su tipi standard - int, char ...)
  - buon funzionamento
  - Supporta OWL DL (parzialmente) e SWRL
  - http://kaon2.semanticweb.org

- Fact++

  - basato su c++
  - free for non commercial use
  - non più aggiornato
  - http://owl.man.ac.uk/factplusplus

Note all'utilizzo di Protege

- Protege è l'editor di ontologie più diffuso (usa ver.3.1.1)

  `http://protege.stanford.edu/download/registered.html`

- Controllare che la porta del ragionatore (in Pellet è stampata all'avvio) sia la stessa di quella dell'editor OWL (Protegé)
  In Protegé si imposta dal menu OWL selezionando preferences
- (ver.3.2.1) Percorso per la creazione di una ontologia che utilizza le description logics
  New Project -> OWL/RDF Files -> Next -> Next -> OWL DL -> Logic View e impostare *DLSintaxClassDisplay* nel menu *OWL->Preferences->ClassDisplayFormat*

Note all'utilizzo di Protege